

**In The United States Patent and Trademark Office
On Appeal From The Examiner To The Board
of Patent Appeals and Interferences**

In re Application of: Anjali Chandnani et al.
Serial No.: 09/905,343
Filing Date: July 14, 2001
Group Art Unit: 2437
Confirmation No. 3770
Examiner: Michael J. Pyzocha
Title: DETECTION OF POLYMORPHIC SCRIPT
LANGUAGE VIRUSES BY DATA DRIVEN
LEXICAL ANALYSIS

MAIL STOP APPEAL BRIEF - PATENTS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

APPEAL BRIEF

Appellants have appealed to the Board of Patent Appeals and Interferences ("Board") from the decision of the Examiner mailed July 21, 2008 (the "Final Office Action"), finally rejecting pending Claims 1-14, 16-22, and 24. Appellants filed a Notice of Appeal on October 21, 2008 along with a Pre-Appeal Brief Request for Review. The Pre-Appeal Panel, in a decision of December 22, 2008, determined that this Application should Proceed to Appeal. Appellants respectfully submit this Appeal Brief with the statutory fee of **\$540.00**.

TABLE OF CONTENTS

	<u>Page</u>
Appeal Brief	1
Table of Contents.....	2
Real Party In Interest	3
Related Appeals and Interferences	4
Status of Claims.....	5
Status of Amendments.....	6
Summary of Claimed Subject Matter	7
Grounds of Rejection to be Reviewed on Appeal	13
Argument	14
I. Claims 1, 7, 8, 10-14, 16, 22, and 24 are patentable under 35 U.S.C. § 103(a) over <i>Chess</i> in view of <i>Yamamoto</i> and <i>Van De Vanter</i>	14
II. Claims 4-6 and 19-21 are patentable under 35 U.S.C. § 103(a) over <i>Chess</i> in view of <i>Yamamoto</i> , <i>Van De Vanter</i> , and <i>Raduchel</i>	16
III. Claim 9 is patentable under 35 U.S.C. § 103(a) over <i>Chess</i> in view of <i>Yamamoto</i> , <i>Van De Vanter</i> , and <i>Bernhard</i>	18
IV. Claims 2, 3, 17, and 18 are patentable under 35 U.S.C. § 103(a) over <i>Chess</i> in view of <i>Yamamoto</i> , <i>Van De Vanter</i> , and <i>Session</i>	19
Conclusion	20
Appendix A: Claims on Appeal.....	21
Appendix B: Evidence.....	28
Appendix C: Related Proceedings.....	29

Real Party In Interest

The real party-in-interest for this Application is Computer Associates Think, Inc., a Delaware corporation, by virtue of a chain of title from the inventors to the current assignee, as shown below:

1. From: Anjali Chandnani
 Trevor Yann
 To: Computer Associates Think, Inc.
 Assignment recorded at Reel/Frame 012573 / 0001,
 on February 11, 2002.

Related Appeals and Interferences

Appellants, the undersigned Attorney for Appellants, and the Assignee know of no applications on appeal that may directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-14, 16-22, and 24 were finally rejected in the Final Office Action dated July 21, 2008. Appellants present Claims 1-14, 16-22, and 24 for appeal and set forth these claims in Appendix A. All claims presented for appeal are shown in Appendix A, attached hereto, along with an indication of the status of those claims.

Status of Amendments

All amendments submitted by Appellants were entered by the Examiner prior to the mailing of the Final Office Action.

Summary of Claimed Subject Matter

With regard to the claims currently under Appeal, Appellants provide the following concise explanation of the subject matter recited in the claim elements. For brevity, Appellants do not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements. Additionally, this explanation should not be used to limit Appellants' claims but is intended to assist the Board in considering the Appeal of this Application.

Embodiments of the present disclosure include tools (in the form of systems and methods) for detecting script language viruses by performing a lexical analysis of a data stream on a computing device/system. The data stream may be generated from a potentially infected file ("subject file"). (*e.g.*, Figures 2-7; Page 6, Lines 22 - 27).

An apparatus 50 for script language virus detection, in accordance with one embodiment shown in Figure 2, comprises script language processor 51, detection data processor 52 and detection engine 53. A methodology for detecting polymorphic script language viruses in accordance with an embodiment of the present disclosure is described with reference to Figures 2 and 3. In this embodiment, language description data corresponding to one or more script languages are prepared by script language processor 51 (step 61). Viral code detection data is prepared by detection data processor 52 for one or more script language viruses (step 62). A data stream is lexically analyzed by detection engine 53 using the language description data and the detection data to detect viral code (step 63). (*e.g.*, Figure 2; Figure 3; Page 7, Lines 14 - 24).

A process for preparing language description data for target script languages is described with reference to Figures 2 and 4. Sets of language definition rules are defined for the respective target script languages and stored in rule base 54 (step 11). Language check rules are defined and also stored in rule base 54 (step 13). Next, the language definition rules and language check rules are processed by script language processor 51 to generate language description data for the respective target script languages (step 15). The language description data for the target script languages are stored in the language description data module 55 (step 17). (*e.g.*, Figure 2; Figure 4; Page 8, Lines 13 - 23).

A process for preparing viral code detection data is described with reference to Figures 2 and 5. Samples of polymorphic script language viral code are collected and stored in code sample store 56 (step 21). The samples are then analyzed by detection data processor 52 (step 23). A detection regimen including layers of token pattern matching and/or CRC signature checking is prepared by the detection data processor 52 (step 25), and the detection regimen is converted to a binary format representing viral code detection data. The viral code detection data is stored in the code detection module 57 (step 27). (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25).

The file to be scanned is converted to a data stream. For example, the data stream may include characters [such as digits in a number (*e.g.*, “0”, “8”, etc.), letters (*e.g.*, “e”, “q”, etc.), symbols (*e.g.*, “+”, “;”, “>”, etc.)] in the file to be scanned. (*e.g.*, Figure 2; Page 11, Lines 16 - 22).

The data stream, in an embodiment in which the target script languages are defined by pattern matching rules and the patterns are associated with output tokens, may be converted to a stream of tokens. The tokens may correspond to respective language constructs, and each token may be a corresponding unique number, symbol, etc. A detection process in that embodiment includes two stages: (i) tokenize the data stream; and (ii) process the tokens using the detection data. (*e.g.*, Figure 2; Page 12, Lines 3 - 8).

A process for tokenizing a data stream will be described with reference to Figures 2 and 6. The detection engine 53 retrieves the language check data from language description module 55 (step 31) and uses the language check data to lexically analyze the data stream to determine the appropriate script language (step 33). The language definition data for the script language determined in step 53 is retrieved from language description module 55 (step 35). Using the language definition data retrieved in step 35, the data stream is again lexically analyzed to generate a stream of tokens (step 37). (*e.g.*, Figure 2; Figure 6; Page 12, Lines 9 - 16).

The data stream corresponding to a file to scan is tokenized by lexical analysis. The data stream is fed to a lexical analyzer in the detection engine which generates a stream of tokens. To tokenize the data stream, a script language used in the data stream is determined using the language check data. The data stream is analyzed using the language check data to

select the language definition data to use for the detection process. Next, the selected language definition data and the data stream are supplied to the lexical analyzer. The data stream is lexically analyzed again, this time using the language definition data, to generate a stream of tokens. Each generated token corresponds to a specific language construct, and may be a corresponding unique number or character. (e.g., Figure 2; Figure 6; Page 12, Lines 17 - 26).

A method for detecting a script language virus is described with reference to Figures 2 and 7. Initially, detection data stored in the code detection module 57 is retrieved (step 41). One of the detection data entries is selected (step 42) and the pattern match or CRC signature check in the selected entry are performed in turn. In step 43, it is determined whether a selected check is a pattern match or virus or CRC signature check. If the check is a pattern match, the token stream is analyzed lexically using the pattern match detection data and language description data (step 44). In step 45, it is determined whether there is a pattern match. If it is determined in step 43 that the check is a CRC signature check, the CRC is run on the token stream (step 48). In step 49, it is determined whether the CRC check succeeds. If the pattern match in step 44 or the CRC check in step 48 is not successful, then the method returns to step 42 to select another detection data entry. If it is successful, detection of viral code is signaled (step 46). (e.g., Figure 2; Figure 7; Page 13, Lines 15 - 26).

With regard to the claims currently under Appeal, Appellants provide the following concise explanation of the subject matter recited in the claim elements. For brevity, ***Appellants do not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements.*** Additionally, this explanation should not be used to limit Appellants' claims but is intended to assist the Board in considering the Appeal of this Application.

For example, Independent Claim 1 recites the following:

A method of detecting script language viruses in data streams (e.g., Figures 2-7; Page 6, Lines 22 - 27) comprising:

preparing language description data corresponding to at least one script language (e.g., Figure 2; Figure 4; Page 8, Lines 13 - 23);

preparing detection data for viral code corresponding to a script language virus (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25);

lexically analyzing a data stream to identify the at least one script language (*e.g.*, Figure 2; Figure 6; Page 12, Lines 9 - 16);

lexically analyzing the data stream using the language description data to generate a stream of tokens (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26); and

lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus (*e.g.*, Figure 2; Figure 7; Page 13, Lines 15 - 26).

As another example, Independent Claim 13 recites the following:

A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting script language viruses (*e.g.*, Figures 2-7; Page 6, Lines 22 - 27), the method steps comprising:

preparing language description data corresponding to at least one script language (*e.g.*, Figure 2; Figure 4; Page 8, Lines 13 - 23);

preparing detection data for viral code corresponding to a script language virus (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25);

lexically analyzing a data stream to identify the at least one script language (*e.g.*, Figure 2; Figure 6; Page 12, Lines 9 - 16);

lexically analyzing the data stream using the language description data to generate a stream of tokens (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26); and

lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus (*e.g.*, Figure 2; Figure 7; Page 13, Lines 15 - 26).

As another example, Independent Claim 14 recites the following:

A computer system, comprising:

a processor; and

a program storage device readable by the computer system, tangibly embodying a program of instructions executable by the processor to perform method steps for detecting script language viruses (*e.g.*, Figures 2-7; Page 6, Lines 22 - 27), the method steps comprising:

preparing language description data corresponding to at least one script language (*e.g.*, Figure 2; Figure 4; Page 8, Lines 13 - 23);

preparing detection data for viral code corresponding to a script language virus (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25); and

lexically analyzing a data stream to identify the at least one script language (*e.g.*, Figure 2; Figure 6; Page 12, Lines 9 - 16);

lexically analyzing the data stream using the language description data to generate a stream of tokens (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26); and

lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus (*e.g.*, Figure 2; Figure 7; Page 13, Lines 15 - 26).

As another example, Independent Claim 16 recites the following:

An apparatus for detecting script language viruses (*e.g.*, Figures 2-7; Page 6, Lines 22 - 27), comprising:

a script language processor, wherein the script language processor prepares language description data corresponding to at least one script language (*e.g.*, Figure 2; Figure 4; Page 8, Lines 13 - 23);

a detection data processor, wherein the detection data processor prepared detection data for viral code corresponding to a script language virus (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25); and

a detection engine, wherein the detection engine converts a data stream to a stream of tokens using lexical analysis (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26), wherein the

tokens correspond to respective language constructs (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26), wherein the detection engine lexically analyzes the stream of tokens using the language description data and the detection data to identify the script language virus (*e.g.*, Figure 2; Figure 7; Page 13, Lines 15 - 26).

As another example, Independent Claim 24 recites the following:

A method (*e.g.*, Figures 2-7; Page 6, Lines 22 - 27), comprising:
receiving a data stream (*e.g.*, Figure 2; Page 11, Lines 16 - 22);
lexically analyzing the data stream to identify a script language (*e.g.*, Figure 2; Figure 6; Page 12, Lines 9 - 16);
receiving language description data for the script language (*e.g.*, Figure 2; Figure 4; Page 8, Lines 13 - 23);
lexically analyzing the data stream using the language description data to generate a stream of tokens (*e.g.*, Figure 2; Figure 6; Page 12, Lines 17 - 26);
generating viral code detection data by analyzing a plurality of samples of polymorphic script language viral code (*e.g.*, Figure 2; Figure 5; Page 10, Lines 12 - 25); and
lexically analyzing the stream of tokens using the viral code detection data and the language description data to identify at least one script language virus (*e.g.*, Figure 2; Figure 7; Page 13, Lines 15 - 26).

Grounds of Rejection to be Reviewed on Appeal

I) Appellants request that the Board review the Examiner's rejection of Claims 1, 7, 8, 10-14, 16, 22, and 24 under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 6,711,583 to Chess et al. ("*Chess*") in view of U.S. Patent No. 5,881,151 to Yamamoto ("*Yamamoto*") and further in view of Van De Vanter ("*Van De Vanter*").

II) Appellants request that the Board review the Examiner's rejection of Claims 4-6 and 19-21 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claims 1 and 16 above, and further in view of U.S. Patent No. 6,418,444 to Raduchel et al. ("*Raduchel*").

III) Appellants request that the Board review the Examiner's rejection of Claim 9 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claim 1 above, and further in view of U.S. Patent No. 6,609,205 to Bernhard et al. ("*Bernhard*").

IV) Appellants request that the Board review the Examiner's rejection of Claims 2, 3, 17, and 18 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claims 1 and 16 above, and further in view of "Session 19: Intro to Complier Design" ("*Session*").

Argument

For at least the following reasons, the Examiner's rejections of Claims 1-14, 16-22, and 24 are improper and should be reversed.

Appellants have made an effort to group claims to reduce the burden on the Board, as contemplated by 37 C.F.R. § 41.37(c)(1)(vii). Where appropriate, Appellants present arguments as to why particular claims subject to a ground of rejection are separately patentable from other claims subject to the same ground of rejection. To reduce the number of groups and thereby reduce the burden on the Board, Appellants do not argue individually every claim that recites patentable distinctions over the references cited by the Examiner, particularly in light of the clear allowability of Appellants' independent claims. The claims of each group provided below may be deemed to stand or fall together for purposes of this Appeal.

Appellants have concluded that the claims may be grouped together as follows:

1. Group 1 may include Claims 1, 7, 8, 10-14, 16, 22, and 24;
2. Group 2 may include Claims 4-6 and 19-21;
3. Group 3 may include Claim 9; and
4. Group 4 may include Claims 2, 3, 17, and 18.

I. Claims 1, 7, 8, 10-14, 16, 22, and 24 are patentable under 35 U.S.C. § 103(a) over Chess in view of Yamamoto and Van De Vanter

The Examiner rejects Claims 1, 7, 8, 10-14, 16, 22, and 24 under 35 U.S.C. § 103(a) as unpatentable over *Chess* in view of *Yamamoto* and further in view of *Van De Vanter*. Appellants respectfully traverse these rejections for at least the reasons provided below.

Claim 24 is patentable at least because the cited references fail to teach or suggest "generating viral code detection data by analyzing a plurality of samples of polymorphic script language viral code." The Final Office Action contends that Column 4, lines 22-61 of *Chess* discloses "preparing detection data for viral code corresponding to the script language virus." See Final Office Action, Pages 2-3. Appellants respectfully disagree. The cited

portion discloses a database 204 of document information. *See Chess*, Column 4, lines 22-26. Database 204 contains the names of documents and program objects within the documents. *See Chess*, Column 4, lines 26-34. *Chess* discloses detecting changes to the documents by comparing the documents to the document database 204. *See Chess*, Column 5, lines 5-38. Even assuming for the sake of argument that document information stored in a database for the purposes of comparison could be characterized as **detection data** (which Appellants do not concede), at no point does *Chess* teach or suggest generating viral code detection data **by analyzing** a plurality of samples of **polymorphic script language viral code**. Indeed, even the Final Office Action concedes that the *Chess-Yamamoto-Van De Vanter* combination fails to explicitly disclose “samples of viral code,” let alone **polymorphic script language viral code**. *See* Final Office Action, Page 6.

Claim 24 is patentable also at least because the cited references fail to teach or suggest “lexically analyzing the stream of tokens using the viral code detection data and the language description data to identify at least one script language virus.” The Final Office Action concedes that *Chess* “fails to explicitly disclose the use of lexical analysis as part of the virus detection.” *See* Final Office Action, Page 3. Instead, the Final Office Action contends that *Yamamoto* discloses “analyzing the stream of tokens using the detection data and the language description data to identify the virus” and *Van De Vanter* discloses “lexically analyzing a stream of tokens.” *See* Final Office Action, Page 3. Appellants respectfully disagree. *Yamamoto* discloses generating tokens and merely **performing a syntax analysis on the tokens**. *See Yamamoto*, Column 4, lines 64-66. For example, *Yamamoto* discloses performing the syntax analysis “to check whether or not each token is present at a grammatically correct position.” *See Yamamoto*, Column 5, lines 11-13. *Yamamoto* does not teach or suggest analyzing the stream of tokens **using viral code detection data** and the language description data. *Van De Vanter* fails to cure this deficiency.

Moreover, the Examiner has failed to establish a *prima facie* case for obviousness, because the Examiner has failed to provide a clear articulation of the reason(s) why the claimed invention would have been obvious to one of skill in the art. The U.S. Supreme Court’s recent decision in *KSR Int’l Co. v. Teleflex, Inc.* reiterated the requirement that Examiners provide an explanation as to why the claimed invention would have been obvious.

KSR Int'l Co. v. Teleflex, Inc., 127 S.Ct. 1727 (2007). The analysis regarding an apparent reason to combine the known elements in the fashion claimed in the patent at issue “should be made explicit.” *Id.* at 1740-41. “Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.” *Id.* at 1741 (internal quotations omitted).

Far from an explicit, articulated reasoning with some rational underpinning, the Examiner simply offers mere conclusory speculation that “it would have been obvious to a person of ordinary skill in the art to use the lexical analysis of a stream of tokens to detect the script viruses of the modified Chess et al. and Yamamoto system. Motivation to do so would have been detect [sic] changes to the stream of tokens.” *See* Final Office Action, Pages 3-4. According to the controlling case law, rules, and guidelines, this type of hindsight analysis, using Appellants’ claims as a roadmap for summarizing references—in this case mischaracterizing references without even citing particular portions—is impermissible. *See* M.P.E.P. § 2142; *KSR*, 127 S.Ct. at 1740-41; *Examination Guidelines*, 72 Fed. Reg. at 57528-29.

For at least those reasons discussed above with regard to Claim 24, Appellants respectfully contend that the cited references do not disclose, teach, or suggest the limitations of Claims 1, 13, 14, and 16. For at least these reasons, Appellants respectfully contend that Claims 1, 13, 14, 16, and 24, and all claims depending therefrom, are patentably distinguishable from the cited references.

II. Claims 4-6 and 19-21 are patentable under 35 U.S.C. § 103(a) over Chess in view of Yamamoto, Van De Vanter, and Raduchel

The Examiner rejects Claims 4-6 and 19-21 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claims 1 and 16 above, and further in view of *Raduchel*. Appellants respectfully traverse these rejections for at least the reasons provided below.

Claim 6 is patentable at least because the cited references fail to teach or suggest “wherein a script language used by the data stream is determined by the lexical analysis using the language check rules.” The Examiner contends that Column 13, lines 4-43 of *Raduchel*

discloses “rule checking.” *See* Final Office Action, Page 5. However, even assuming for the sake of argument that *Raduchel* discloses rule checking (which Appellants does not concede), rule checking does not teach or suggest **determining a script language using language check rules**. *Chess, Yamamoto, and Van De Vanter* fail to cure this deficiency.

Moreover, the Examiner’s attempt to combine *Raduchel* with an alleged combination of **three other references** (*Chess, Yamamoto* and *Van De Vanter*) is improper because the Examiner failed to establish a *prima facie* case for obviousness. Again, far from an explicit, articulated reasoning with some rational underpinning, the Examiner offers only the following conclusory statement for making the *Chess, Yamamoto-Van De Vanter-Raduchel* combination: “[m]otivation to do so would have been prevent [sic] programs that can pose risks to a computer system because they do not conform to the language standard (see column 1 lines 39-52).” Appellants respectfully submit that this is incorrect. Appellants contend that the Examiner mischaracterized the cited portion because it fails to provide a suggestion of any sort. The entirety of Column 1, lines 39-52 of *Raduchel* states:

A program that does not conform to the Pure Java standard can pose risks to a computer system. For example, such a program may allow computer viruses to infiltrate the system. As an example, the Java language generally allows the execution of “native” code (i.e., code written in a programming language other than Java) from within a Java program. Such native code can include a virus program capable of escaping from the Java runtime environment into other portions of the computer memory and storage media. Such viruses could corrupt a user’s hard drive, overflow the program stack, or perform similar types of damage, either intentionally or unintentionally. It is desirable to ensure that such damage does not occur. Thus, in the Pure Java standard, the user is not allowed to write calls to native methods.

Appellants submit that this type of hindsight analysis, using Appellants’ claims as a roadmap for summarizing references—in this case relying solely on one mischaracterized portion of a reference to combine it with three other references—is impermissible.

Finally, Appellants submit that these claims are patentable at least because these claims depend from one of the patentable independent claims discussed above. For at least

these reasons, Appellants respectfully contend that Claims 4-6 and 19-21 are patentably distinguishable from the cited references.

III. Claim 9 is patentable under 35 U.S.C. § 103(a) over Chess in view of Yamamoto, Van De Vanter, and Bernhard

The Examiner rejects Claim 9 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claim 1 above, and further in view of *Bernhard*. Appellants respectfully traverse these rejections for at least the reasons provided below.

Claim 9 is patentable at least because the cited references fail to teach or suggest “setting a detection regimen that includes at least one pattern match or cyclical redundancy check based on the analysis of the obtained samples.” The Examiner concedes that the modified *Chess*, *Yamamoto* and *Van De Vanter* system does not disclose this limitation and contends that Column 2, lines 14-28 of *Bernhard* discloses this limitation. *See* Final Office Action, Page 6. Appellants respectfully disagree. The cited portion of *Bernhard* discloses detecting signatures. *See Bernhard*, Column 2, lines 14-18. At no point does *Bernhard* teach or suggest **setting a detection regimen** based on the analysis of obtained samples. *Chess*, *Yamamoto*, and *Van De Vanter* fail to cure this deficiency.

Moreover, the Examiner’s attempt to combine *Bernhard* with an alleged combination of **three other references** (*Chess*, *Yamamoto* and *Van De Vanter*) is improper because the Examiner failed to establish a *prima facie* case for obviousness. Again, far from an explicit, articulated reasoning with some rational underpinning, the Examiner offers only the following conclusory statement for making the *Chess*, *Yamamoto-Van De Vanter-Bernhard* combination: “[m]otivation to do so would have been to consolidate the reference signatures (see column 2 lines 34-45).” Appellants respectfully submit that this is incorrect. Appellants contend that the Examiner mischaracterized the cited portion because it fails to provide a suggestion of any sort. The entirety of Column 2, lines 34-45 of *Bernhard* states:

An advantage of the invention is that an entire set of reference signatures can be consolidated into a smaller set of decision graphs. The use of a single decision graph for multiple signatures results in more efficient processing. Signatures having common events can be identified, and decision graphs

optimized so that they represent those signatures that are most efficiently processed together. Each common event in a decision graph results in one matching step for that event rather than one matching step for each signature. Depending on the number of common events that are processed together, the overall reduction in the processing load can be substantial.

Appellants submit that this type of hindsight analysis, using Appellants' claims as a roadmap for summarizing references—in this case relying solely on one mischaracterized portion of a reference to combine it with three other references—is impermissible.

Finally, Appellants submit that Claim 9 is patentable at least because it depends from patentable independent Claim 1 discussed above. For at least these reasons, Appellants respectfully contend that Claim 9 is patentably distinguishable from the cited references.

IV. Claims 2, 3, 17, and 18 are patentable under 35 U.S.C. § 103(a) over Chess in view of Yamamoto, Van De Vanter, and Session

The Examiner rejects Claims 2, 3, 17, and 18 under 35 U.S.C. § 103(a) as unpatentable over the modified *Chess*, *Yamamoto* and *Van De Vanter* system as applied to Claims 1 and 16 above, and further in view of *Session*. Appellants respectfully traverse these rejections for at least the reasons provided below.

Appellants submit that these claims are patentable at least because these claims depend from one of the patentable independent claims discussed above. For at least these reasons, Appellants respectfully contend that Claims 2, 3, 17, and 18 are patentably distinguishable from the cited references.

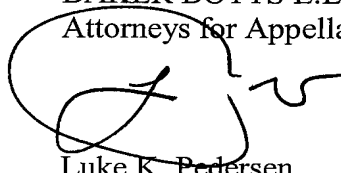
Conclusion

Appellants have demonstrated that the present invention, as claimed, is clearly distinguishable over the prior art cited by the Examiner. Therefore, Appellants respectfully request the Board of Patent Appeals and Interferences to reverse the Examiner's final rejection of the pending claims and instruct the Examiner to issue a notice of allowance of all pending claims.

The Commissioner is hereby authorized to charge the large entity fee of **\$540.00** under 37 C.F.R. § 41.20(b)(2) and to charge any additional fees or credit any overpayments to **Deposit Account No. 02-0384 of Baker Botts L.L.P.**

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellants

A handwritten signature in black ink, appearing to be 'Luke K. Pedersen', is written over the printed name and is partially enclosed by a circular stamp.

Luke K. Pedersen
Reg. No. 45,003
Tel. (214) 953-6655

Date: 1/21/09

CORRESPONDENCE ADDRESS:

Customer No. **05073**

Appendix A: Claims on Appeal

1. **(Previously Presented)** A method of detecting script language viruses in data streams comprising:

preparing language description data corresponding to at least one script language;
preparing detection data for viral code corresponding to a script language virus;
lexically analyzing a data stream to identify the at least one script language;
lexically analyzing the data stream using the language description data to generate a stream of tokens; and
lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus.

2. **(Original)** The method of claim 1, wherein the language description data correspond to Dynamic Finite Automata data.

3. **(Original)** The method of claim 2, wherein the Dynamic Finite Automata data comprises a set of states, with each state having a corresponding set of transaction having an associated character to be matched and an associated next state.

4. **(Previously Presented)** The method of claim 1, wherein the language description data correspond to language definition rules and check rules, wherein the language definition rules include descriptions of constructs of the target script language and relationships between the constructs.

5. **(Original)** The method of claim 4, wherein the lexical analysis includes one or more pattern matches based on the language definition rules.

6. **(Original)** The method of claim 4, wherein a script language used by the data stream is determined by the lexical analysis using the language check rules.

7. **(Original)** The method of claim 1 further comprising setting language definition rules for each of the least one script language.

8. **(Original)** The method of claim 1, wherein the detection data comprise at least one test, wherein each of the at least one test correspond to a pattern match or a cyclical redundancy check.

9. **(Original)** The method of claim 1, wherein the step of preparing detection data comprises:

obtaining samples of the viral code;
analyzing the obtained samples; and
setting a detection regimen that includes at least one pattern match or cyclical redundancy check based on the analysis of the obtained samples.

10. **(Original)** The method of claim 1, wherein the data stream is converted to a stream of tokens using lexical analysis.

11. **(Original)** The method of claim 10, wherein the tokens correspond to respective language constructs.

12. **(Original)** The method of claim 10, wherein a cyclical redundancy check is performed on the stream of tokens to detect viral code.

13. **(Previously Presented)** A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting script language viruses, the method steps comprising:

preparing language description data corresponding to at least one script language;

preparing detection data for viral code corresponding to a script language virus;

lexically analyzing a data stream to identify the at least one script language;

lexically analyzing the data stream using the language description data to generate a stream of tokens; and

lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus.

14. **(Previously Presented)** A computer system, comprising:
a processor; and
a program storage device readable by the computer system, tangibly embodying a program of instructions executable by the processor to perform method steps for detecting script language viruses, the method steps comprising:
preparing language description data corresponding to at least one script language;
preparing detection data for viral code corresponding to a script language virus; and
lexically analyzing a data stream to identify the at least one script language;
lexically analyzing the data stream using the language description data to generate a stream of tokens; and
lexically analyzing the stream of tokens using the detection data and the language description data to identify the script language virus.

15. **(Cancelled)**

16. **(Previously Presented)** An apparatus for detecting script language viruses, comprising:

a script language processor, wherein the script language processor prepares language description data corresponding to at least one script language;

a detection data processor, wherein the detection data processor prepared detection data for viral code corresponding to a script language virus; and

a detection engine, wherein the detection engine converts a data stream to a stream of tokens using lexical analysis, wherein the tokens correspond to respective language constructs, wherein the detection engine lexically analyzes the stream of tokens using the language description data and the detection data to identify the script language virus.

17. **(Original)** The apparatus of claim 16, wherein the language description data correspond to Dynamic Finite Automata data.

18. **(Original)** The apparatus of claim 17, wherein the Dynamic Finite Automata data comprises at least one set of states, with each state having a corresponding set of transitions and each transition having an associated character to be matched and an associated next state.

19. **(Previously Presented)** The apparatus of claim 16, wherein the language description data corresponds to language definition rules and language check rules, wherein the language definition rules include descriptions of constructs of the target script language and relationships between the constructs.

20. **(Original)** The apparatus of claim 19, wherein the lexical analysis by the detection engine includes one or more pattern matches based on the language definition rules.

21. **(Original)** The apparatus of claim 19, wherein a script language used by the data stream is determined by the lexical analysis of the detection engine using the language check rules.

22. **(Original)** The apparatus of claim 16, wherein the detection data comprises at least one test, and each of the at least one test correspond to a pattern match or a cyclical redundancy check.

23. **(Cancelled)**

24. **(Previously Presented)** A method, comprising:
receiving a data stream;
lexically analyzing the data stream to identify a script language;
receiving language description data for the script language;
lexically analyzing the data stream using the language description data to generate a
stream of tokens;
generating viral code detection data by analyzing a plurality of samples of
polymorphic script language viral code; and
lexically analyzing the stream of tokens using the viral code detection data and the
language description data to identify at least one script language virus.

25. **(Cancelled)**

26. **(Cancelled)**

Appendix B: Evidence

(NONE)

Appendix C: Related Proceedings

The Appellants, the Attorney for Appellants, and the Assignee know of no applications on appeal that may directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.